

## Mass & Spring Systems: The Wave Equation

Mass and spring systems also are dynamic systems. One way to model these systems is to

- 1) Replace the mass, spring, and friction terms with their LaPlace admittance,
- 2) Redraw the system as an electric circuit, and
- 3) Write the voltage node equations.

The LaPlace admittance's come from modeling the system as

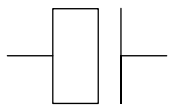

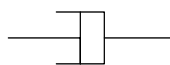
$$\text{Force} = \text{mass} * \text{acceleration}$$

$$F = Z X \quad (F = \text{force}, Z = \text{admittance}, X = \text{displacement})$$

which has the electrical analog

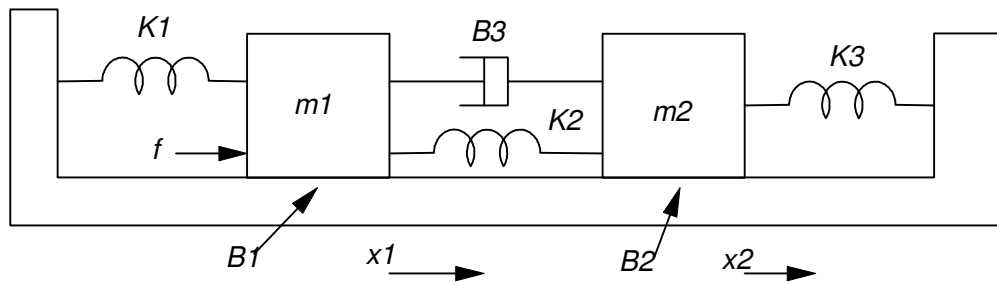
$$I = G V \quad (\text{current} = \text{admittance} * \text{voltage}).$$

Mechanical System	Electrical Analog
Force in the positive direction	Current to the node
Displacement in the positive direction	Positive voltage
Mass, (see below)	Admittance

	Symbol	F = Z X Relationship	LaPlace value of Admittance
Mass		$f = m x''$	$s^2 m$
Spring		$f = k x$	k
Friction		$f = B x'$ $f = f_v x'$	sB s f <sub>v</sub>

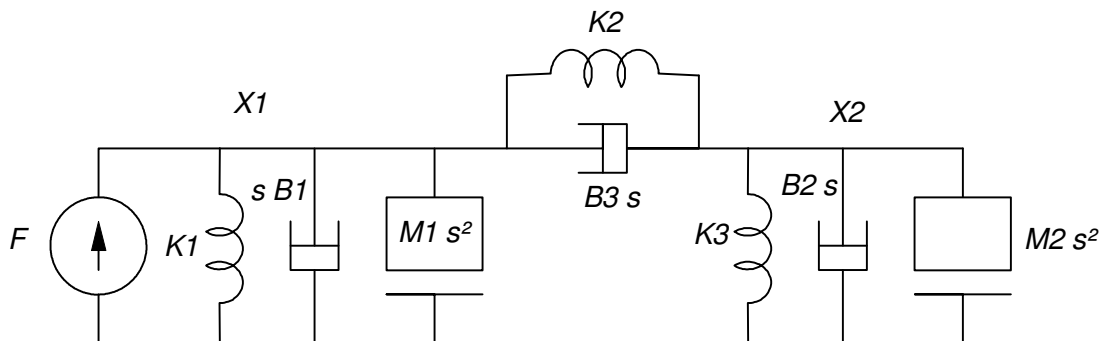
**Example:**

Write the equations of motion for the following mass and spring system:



**Solution:**

Step 1: Draw the circuit equivalent:



Step 2: Write the voltage node equations. From before

(The sum of the admittance's to node a)Va - (the sum of the admittances from node a to b)Vb - ...  
 = (The current to node a)multiply a: times (1/R2) and

$$(K_1 + B_1s + M_1s^2 + K_2 + B_3s)X_1 - (K_2 + B_3s)X_2 = F$$

$$(M_2s^2 + B_2s + K_3 + K_2 + B_3s)X_2 - (K_2 + B_3s)X_1 = 0$$

Step 3: Solve to find the output as a function of the input. State-Space really helps here

Solve for the highest derivative:

$$M_1 s^2 X_1 = -(K_1 + K_2 + B_1 s + B_3 s) X_1 + (K_2 + B_3 s) X_2 + F$$

$$M_2 s^2 X_2 = -(B_2 s + K_3 + K_2 + B_3 s) X_2 + (K_2 + B_3 s) X_1$$

Put in matrix form. Note that there are two energy states for each mass

- Position (X)
- Velocity (sX)

$$s \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ sX_1 \\ sX_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \vdots & 1 & 0 \\ 0 & 0 & \vdots & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots \\ \left(\frac{-(K_1+K_2)}{M_1}\right) & \left(\frac{K_2}{M_1}\right) & \vdots & \left(\frac{-(B_1+B_3)}{M_1}\right) & \left(\frac{B_3}{M_1}\right) \\ \left(\frac{K_2}{M_2}\right) & \left(\frac{-(K_2+K_3)}{M_2}\right) & \vdots & \left(\frac{B_3}{M_2}\right) & \left(\frac{-(B_2+B_3)}{M_2}\right) \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ sX_1 \\ sX_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dots \\ \left(\frac{1}{M_1}\right) \\ 0 \end{bmatrix} F$$

$$Y = X_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ sX_1 \\ sX_2 \end{bmatrix} + [0]F$$

Note that

- You have 2N states, where N is the number of masses. Each mass has two energy states (kinetic and potential energy) giving your 2N state variables.
- The first N rows are [ 0 : I ] where I is the identity matrix. This tells MATLAB that the states are position and velocity.
- The last N rows are where the dynamics come into play.

Also also, you can have real or complex poles for mass-spring systems - unlike the heat equation which always has real poles.

**Finding the Transfer Function:**

To find the transfer function, use MATLAB or SciLab. Assume for example that

- M = 1kg
- B = 2 Ns/m
- K = 10 N/m

Then the state-space model is:

$$s \begin{bmatrix} X_1 \\ X_2 \\ sX_1 \\ sX_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -20 & 10 & -4 & 2 \\ 10 & -20 & 2 & -4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ sX_1 \\ sX_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} F$$

**MATLAB Code:** Input the A B C D matrices:

```
A = [0,0,1,0 ; 0,0,0,1 ; -20,10,-4,2 ; 10,-20,2,-4];
B = [0;0;1;0];
C = [0,1,0,0];
D = 0;
```

Use these to define G(s):

```
G = ss(A,B,C,D)
```

Once G(s) is in MATLAB, find the transfer function

```
zpk(G)
```

$$\frac{2(s+5)}{(s^2 + 2s + 10)(s^2 + 6s + 30)}$$

meaning:

$$X_2 = \left( \frac{2(s+5)}{(s^2+2s+10)(s^2+6s+30)} \right) F$$

If you want to approximate this with a 2nd-order model, keep the slowest pole and match the DC gain

```
DC = evalfr(G,0)
0.0333333
```

So

$$X_2 \approx \left( \frac{0.3333}{(s+1 \pm j3)} \right) F$$

To check the response in MATLAB, take the two step responses.

Input the 2nd-order approximation:

```
G2 = zpk([], [-1+j*3, -1-j*3], 0.3333)
```

---

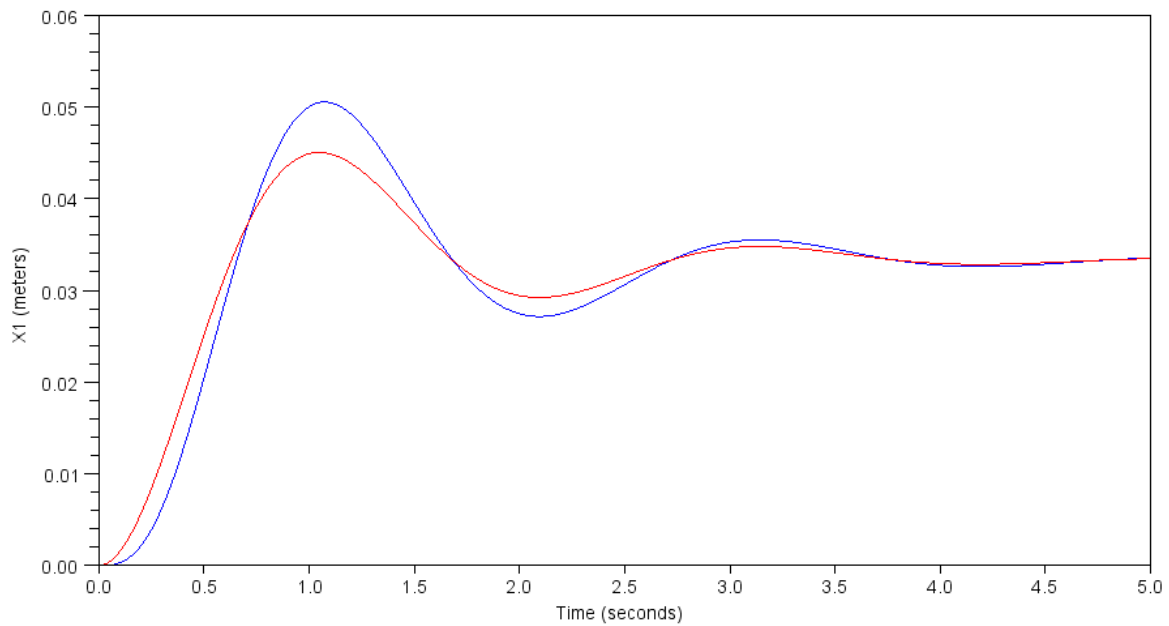
Take the step response of the two systems:

```
t = [0:0.1:100]';  
x2 = step(G,t);  
x2a = step(G2,t);
```

and plot

```
plot(t,x2,t,x2a)  
xlabel('Time (seconds)');  
ylabel('X2 (meters)');
```

Note that the 2nd-order model isn't that good of an approximation: the 'fast' pole is only 3x faster.



## Matlab Code:

```

a11 = zeros(2,2);
a12 = eye(2,2);
a21 = [-20,10;10,-20];
a22 = [-4,2;2,-4];
A = [a11,a12;a21,a22]

```

```

    0    0    1    0
    0    0    0    1
   -20   10   -4    2
    10   -20    2   -4

```

```

B = [0;0;1;0];
C = [0,1,0,0];
D = 0;
G = ss(A,B,C,D);

```

```
zpk(G)
```

```

          2 (s+5)
-----
(s^2 + 2s + 10) (s^2 + 6s + 30)

```

```
tf(G)
```

```

          2 s + 10
-----
s^4 + 8 s^3 + 52 s^2 + 120 s + 300

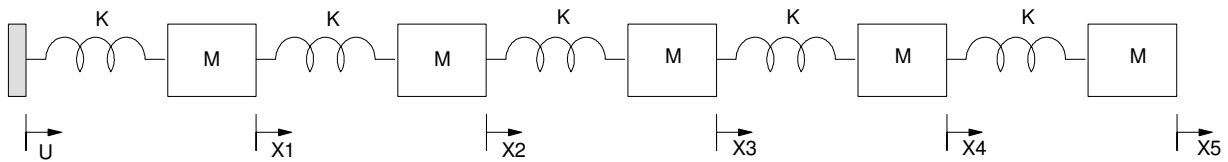
```

$$X_2 = \left( \frac{2(s+5)}{(s+3+j4.58)(s+1\pm j3)} \right) F$$

## Wave Equation: (fun stuff)

If you have  $N$  mass-spring systems in series, you get coupled 2nd-order differential equations, termed the Wave equation

$$\frac{d^2 x_i}{dt^2} = f(x_{i-1}, x_i, x_{i+1})$$



Cascaded Mass-Spring Systems creates the Wave equation

This is almost exactly like the heat equation. When you write the node equation for mass 2, you get

$$Ms^2 x_2 = Kx_1 - 2Kx_2 + Kx_3$$

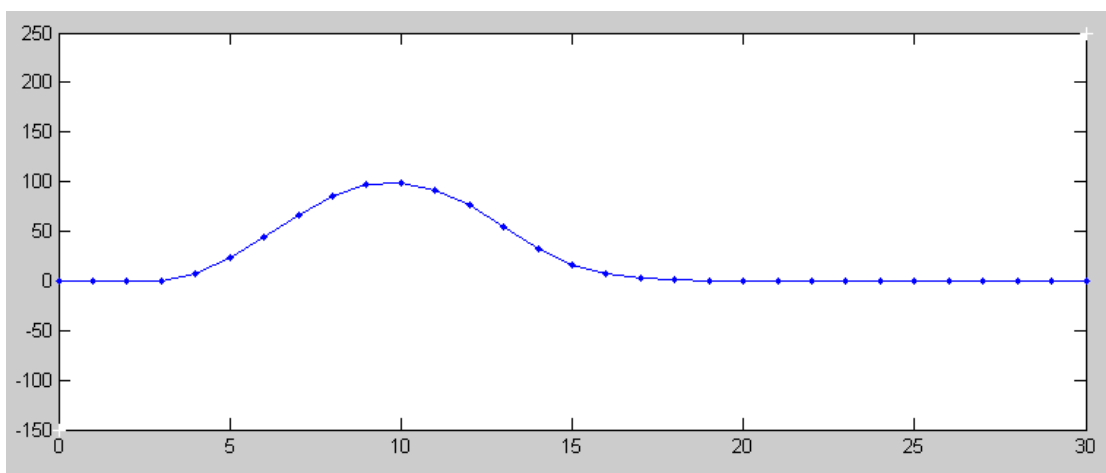
$$s^2 x_2 = \left(\frac{K}{M}\right)x_1 - \left(\frac{2K}{M}\right)x_2 + \left(\frac{K}{M}\right)x_3$$

which is exactly the same form as we got for the heat equation, except that the result is the second derivative of  $x_2$  rather than the first derivative. This has a profound impact on how the system behaves.

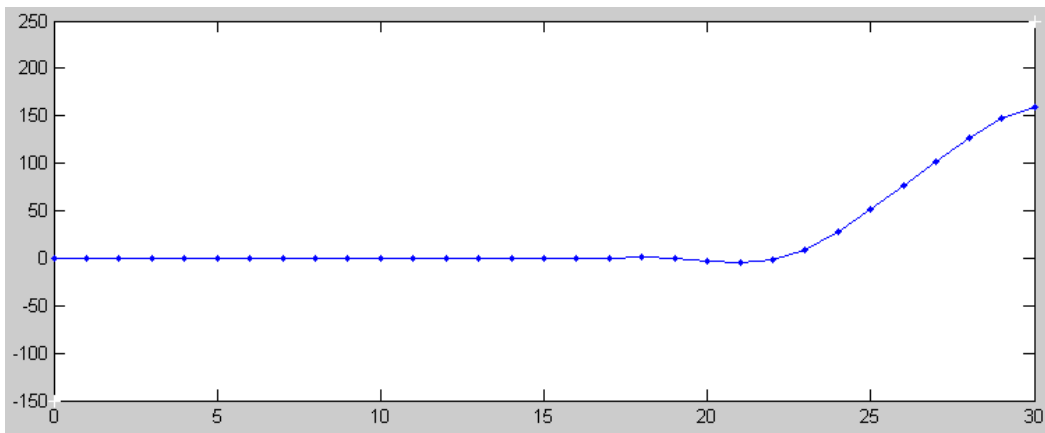
For example, assume you have

- 30 nodes (30 finite elements) and
- $K/M = 50$
- Friction = 0.01 at each node

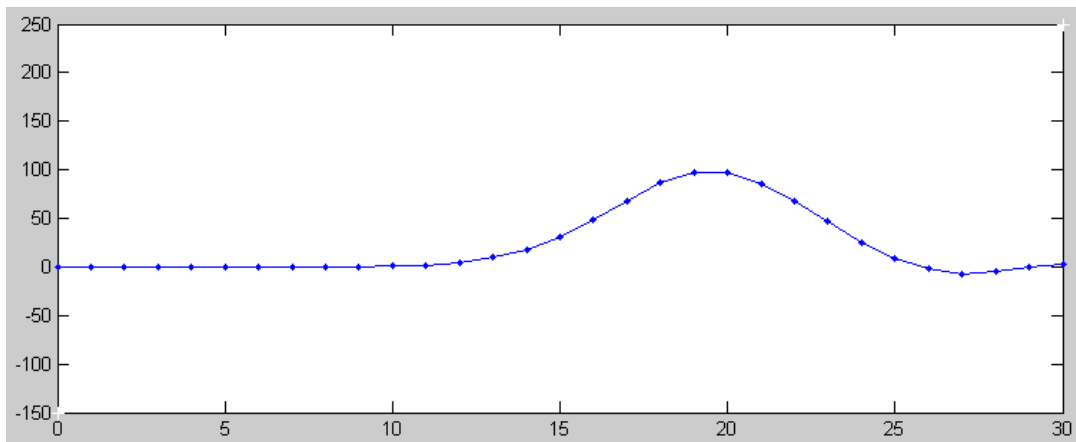
If you flick the input ( $x(0)$ ) to launch a wave, then at  $t=2$  seconds, the states look like the following:



$t = 2$  seconds. Wave traveling to the right



t = 5 seconds. Wave hits the right endpoint



t = 7 seconds. Reflection is now traveling to the left

In terms of a transfer function, this is really ugly.

- There are 30 finite elements, each with two energy states (kinetic and potential energy)
- The resulting system is 60th order
- All 60 poles are on the  $j\omega$  axis - all 60 are dominant.

If you did find the transfer function and plot the step response, you would see how the right endpoint responds ( $x(30)$ ). If you run the following simulation, you see how all 30 points respond.



---

**Wave.m**

```
N = 30;    % number of nodes

V = zeros(N,1);
dV = zeros(N,1);

t = 0;
dt = 0.01;

while(t < 100)

    if (t < 2) V0 = 100 * ( ( sin(0.5*pi*t) ) ^2 );
        else V0 = 0;
        end

    ddV(1) = 50*V0 - 100*V(1) + 50*V(2) - 0.01*dV(1);

    for i=2:N-1
        ddV(i) = 50*V(i-1) - 100*V(i) + 50*V(i+1) - 0.01*dV(i);
    end

    ddV(N) = 50*V(N-1) - 50*V(N) - 0.01*dV(N);

    for i=1:N
        dV(i) = dV(i) + ddV(i)*dt;
        V(i) = V(i) + dV(i)*dt;
    end
    t = t + dt;

    plot([0:N],[V0;V],'.-');
    ylim([-100,150]);
    pause(0.01);

end
```